Covid on Campus:
Simulating the Testing and Testing the Simulation

Hope Neveux
Hope.Neveux1@Marist.edu

Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu

NYCWiC — April 2022

Background

Simulating the Testing

Testing the Simulation

Background

Simulating the Testing

Testing the Simulation

Background

Summer 2020

The Covid-19 pandemic presented a vast array of challenges.

One challenge: Efficiently testing large populations

- Bad news: Complex, difficult, time-consuming, critical
- Good news: A fun project and a great motivating example for coursework

Plan for that Fall semester:

- Generate representative samples of our 6000-person community
- Produce testing invites, track responses and results
- Pooled surveillance testing

  and . . .
- Make use of all this in the classroom

Plan for the following Spring semester:

- Test everybody every two weeks

Background — Representative Samples

Summer 2020

How to generate representative samples?

- Capacity is limited by test kits, testing staff, testing hours, and throughput
- Advice from MIPO: Don't be too granular.

Our plan:

- On-campus students: (pseudo-) randomly select 17%-30% of the population each day Mon—Sat, grouped by dormitories
- Off-campus students: select 2.4% per day, Mon—Fri
- Faculty and staff: select 1% per day, Mon—Fri

Who gets selected?

- We choose people for each day who have…
  - not been tested in the prior two weeks
  - not previously tested positive
  - not been excluded (for being fully remote, in quarantine, doing an off-site internship, part of a traveling sports team, etc.)

C19 Testing Plan :: Clusters

Cluster	Building	Fall 2020 tracked occupancy	Number to Select	Approximate Test Density *
Monday	Champagnat	394	79	20.1%
	Midrise	238	48	20.2%
	off-campus Students	N/A	42	2.4%
	off-campus Employees	N/A	11	1.0%
		632	**180**	
Tuesday	Leo	282	72	25.5%
	Marian	111	28	25.2%
	Sheahan	105	27	25.7%
	off-campus Students	N/A	42	2.4%
	off-campus Employees	N/A	11	1.0%
		498	**180**	
Wednesday	DCC Conklin	116	34	29.3%
	Foy	195	57	29.2%
	New Townhouses	125	36	28.8%
	off-campus Students	N/A	42	2.4%
	off-campus Employees	N/A	11	1.0%
		436	**180**	
Thursday	Fulton	222	60	27.0%
	New Fulton	252	67	26.6%
	off-campus Students	N/A	42	2.4%
	off-campus Employees	N/A	11	1.0%
		474	**180**	
Friday	Lower West	213	61	28.6%
	Upper West	233	66	28.3%
	off-campus Students	N/A	42	2.4%
	off-campus Employees	N/A	11	1.0%
		446	**180**	
Saturday	Lavelle	165	28	17.0%
	McCormick	146	25	17.1%
	O Shea	159	27	17.0%
	Ward	280	47	16.8%
	off-campus Students	N/A	42	2.4%
	off-campus Employees	N/A	11	1.0%
		750	**180**	
On-campus student population		**3236**	1080	17.73%
* Extra on-campus students in selection			0	per day
Off-campus Employees		1116	11	per day
Off-campus students (remote/commuter)		1738	42	per day
Size of off-campus population		**2854**		
Total population		**6090**		

Background — Invites, Responses, and Results

Summer 2020

PostgreSQL Database

- for generating and scheduling testing invitations
- for tracking compliance
- for tracking results

```
create table People (
   pid                 text,
   isStudent           boolean,
   firstName           text,
   lastName            text,
   major               text,
   email1              text,
   email2              text,
   phone1              text,
   phone2              text,
   livesOnCampus       boolean,
   dormBuilding        text,
   dormFloor           int,
   dormRoom            text,
   dormBuildingDesc    text,
   isEmployee          boolean,
   empDept             text,
   empTitle            text,
   empPhone            text,
   selectedForTesting  date,
   -- alreadyTested     boolean added  9/11/2020
   -- invitedForTesting date    added  9/12/2020
   -- didComply         boolean added  9/28/2020
   -- FTE               text    added 10/23/2020
   -- isRemoteEmp       bool    added 10/23/2020
  primary key (pid)
);
```

(a) Student entity subtype attributes

(b) Employee entity subtype attributes

(c)

```
-- Asymptomatic surveillance COVID screening
-- The PostgreSQL random() function is based on the POSIX-standard erand48() family.
-- It's not good enough for cryptography, but it's plenty good for us.
create or replace function choosePeopleToTest()
   returns table (
        ...
    )
language plpgsql
as
$$
declare
   currentCluster       text;
   clusterRow           record;
   dormRow              record;
   goBackAtLeastThisFar date;
begin
   -- Establish the date prior to which we'll consider re-inviting people for testing.
   goBackAtLeastThisFar := (current_date - interval '13 days');
   for clusterRow in ( select distinct dayNameToDayNumber(d.cluster), d.cluster
                       from Dorms d
                       order by dayNameToDayNumber(d.cluster) ) loop
      currentCluster := clusterRow.cluster;
      for dormRow in (select building, numToTest
                      from Dorms d
                      where d.cluster = currentCluster
                      order by d.building) loop
         insert into workTable
         select p.pid,
                ...
         from People p inner join Dorms d on p.dormBuilding = d.building
         where p.livesOnCampus
           and p.isStudent
           and ( (p.selectedForTesting is null)
                  or
                 (p.selectedForTesting < goBackAtLeastThisFar) )
           and p.dormBuilding = dormRow.building
         order by previouslyTested ASC, random()
         limit dormRow.numToTest;
      end loop; -- for dormRow
      -- This completes one cluster. Loop to the next.
   end loop; -- for clusterRow
   -- That's it. We're done. Return the results.
   return query (select * from workTable);
end;
$$
```

Background — Pooled Testing

Summer 1941

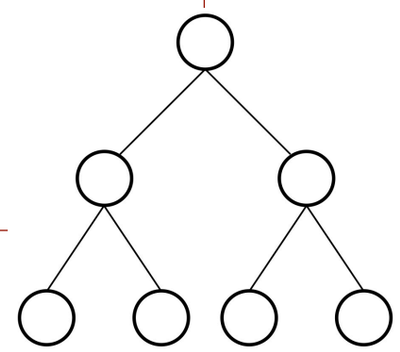
WWII-era screening strategy

- US Army economists Robert Dorfman and David Rosenblatt created a group testing method for detecting (and rejecting) syphilitic draft candidates.
- Combining biomarkers (spit) from multiple people into single test (tube) can reduce the overall number of tests needed, so long as…
  - we have an unbiased and uniform population to test, and
  - the test is sufficiently accurate, sensitive, and specific to Covid-19

A testing protocol:

Start with groups of size $g \in \{4, 8, 16, 32, 64\}$

```
test g elements
if infection found
   divide into two equal size groups and test each group
   if one group shows infection and the other does not
      test all members of the infected group and clear the others // done with 1 + 2 + g/2 tests (the rest of the cases)
   else // both groups show infection
      test all members of both groups // done with 1 + 2 + g tests (worst case)
   end if
else
  // done with 1 test (best case)
end if
```



Background — Covid in the classroom

Fall 2020

Everybody was living the same Covid experience. We could all relate to it, no matter the topic.

- Operating Systems
  - bounded buffer producer-consumer problem
- Database Systems
  - database design
  - SQL, joins, subqueries
  - stored procedures
- Algorithms
  - simulating testing protocols

Algorithms

Semester Project - 100 points

Goals

- to have a semester-long programming project that you can work on a little bit each week.
- to develop a simulation of group/pooled testing.

Requirements and Notes

- Read the article on our class web site about group/pooled testing.
- Program a simulation of the testing protocol described in the article. Run your simulation on population of 1000 people (at first) in groups of 8 assuming a 2% infection rate and 100% accurate tests. Output the results in a manner similar to those shown below.

Using the protocol described in the article, there are three possibilities to consider:

(1) there are no infected samples
(2) there is exactly one infected sample
(3) there are two or more infected samples

We can determine the likelihood of each possibility based on the number of samples we pool into each group and the infection rate of the disease. For details of those percentages and how and we can derive them, read the article again.

Program your simulation to test groups of 8 for a disease with an infection rate of 2% . For 1000 people (where 20 of them (2%) are infected and 980 are infection-free), you would make 125 groups of 8 samples each and simulate the tests. Based on the expected values (see the article) we expect the results to be as follows:

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

The output of your simulation should match or be very close to those values. Once you have that working, try it with 10,000 and 100,000 and 1M people, still in groups of 8, still with a 2% infection rate, still with 100% accurate tests. The larger the population you test, the closer you should get to the statistically expected values.

Document the results of your simulation for the various population sizes with LaTeX and commit that to your GitHub repository along with your code.

Resources

- Everything we're doing this semester is likely to be of use for you in this project.

Hints

Do not just infect the population and then count the cases. You must actually simulate the testing protocol and count the tests used by case along the way.

Start early. Do a little bit every day.

Submitting Your Work

Commit your final work in your private GitHub repository on or before the due date (see our syllabus).

Background

Simulating the Testing

Testing the Simulation

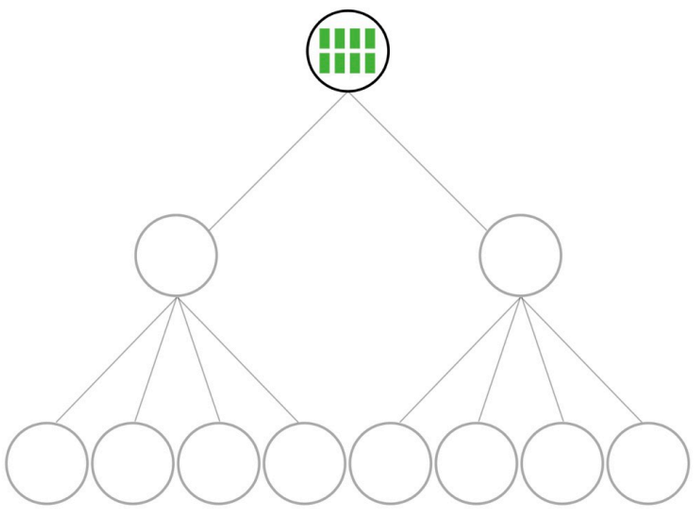
Simulating the Testing

A Simple Analysis

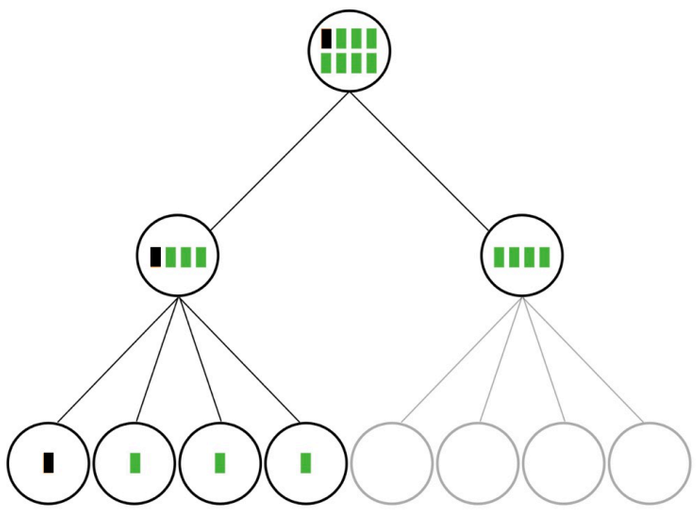
Start with groups of size $g \in \{4, 8, 16, 32, 64\}$

```
test g elements
if infection found
   divide into two equal size groups and test each group
   if one group shows infection and the other does not
      test all members of the infected group and clear the others // done with 1 + 2 + g/2 tests (the rest of the cases)
   else // both groups show infection
      test all members of both groups // done with 1 + 2 + g tests (worst case)
   end if
else
  // done with 1 test (best case)
end if
```

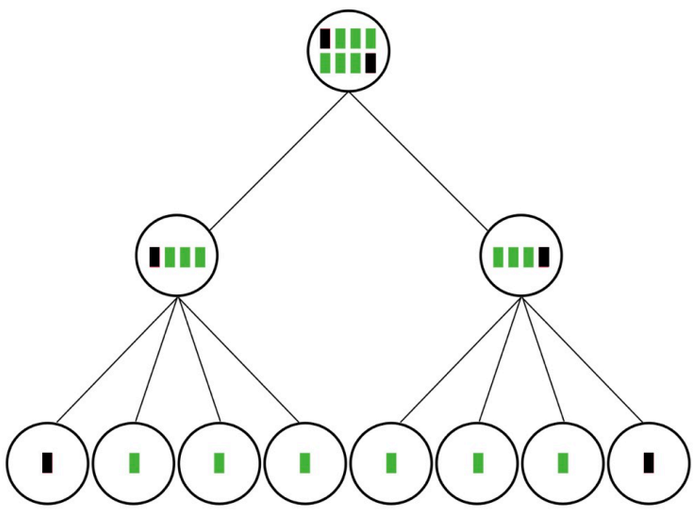
Given the protocol above with group size = 8, there are three possibilities to consider:
(1) There are no infected samples.
(2) There is exactly one infected sample.
(3) There are two or more infected samples.



Case (1) : no infections, 1 test



Case (2) : 1 infection, 7 tests



Case (3) : 2+ infections, 11 tests (worst case)

Simulating the Testing

A Simple Analysis

For an infection rate of 2%:

(1) There are no infected samples.

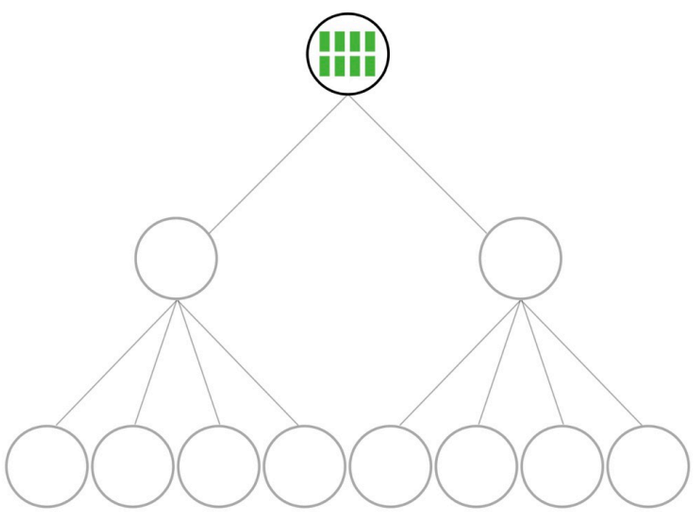
This case is expected to happen 85% of the time. This is because a 2% infection rate means that, on average, 98% of the population is uninfected. The likelihood of randomly choosing 8 uninfected samples is 0.98^8, which is 0.85. When this occurs only one test is needed.

(2) There is exactly one infected sample.

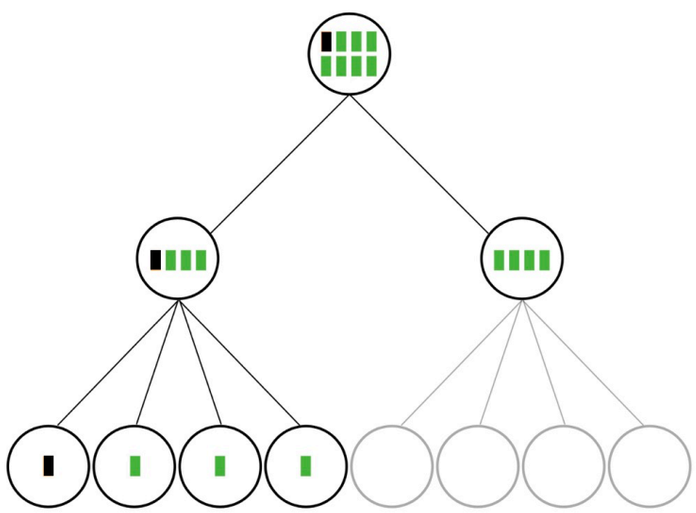
This case is the only other possibility, happens the rest of the time, which is 14.96%, and 7 tests are needed.

(3) There are two or more infected samples.

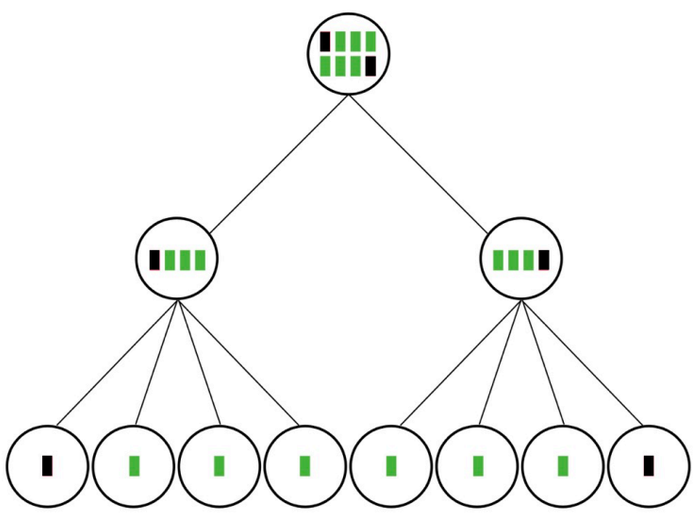
This case happens slightly less than 0.04% of the time because the likelihood of randomly choosing two infected samples is 0.02 × 0.02, or 0.0004 or 0.04%. (It's actually less, but it's safe to err on the side of an upper bound value. Also, the likelihood of randomly choosing more than two infected samples is even lower, so again we are safe with this upper bound.) In this case, 11 tests are needed.



Case (1) : no infections, 1 test



Case (2) : 1 infection, 7 tests



Case (3) : 2+ infections, 11 tests (worst case)

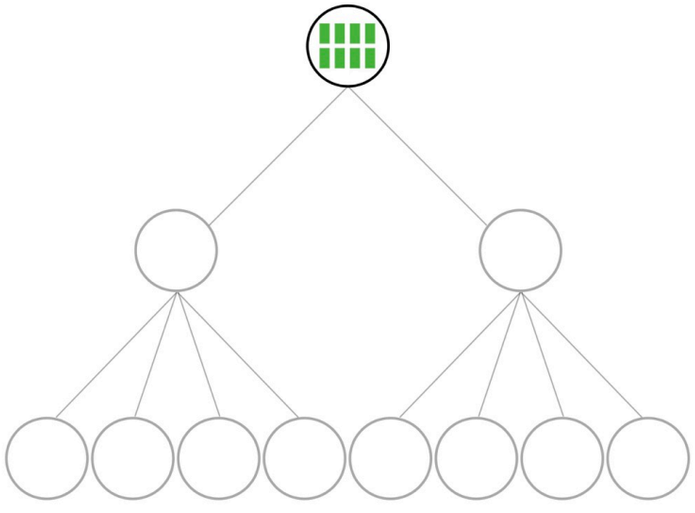
Simulating the Testing

A Semester-long Project in Java or C++

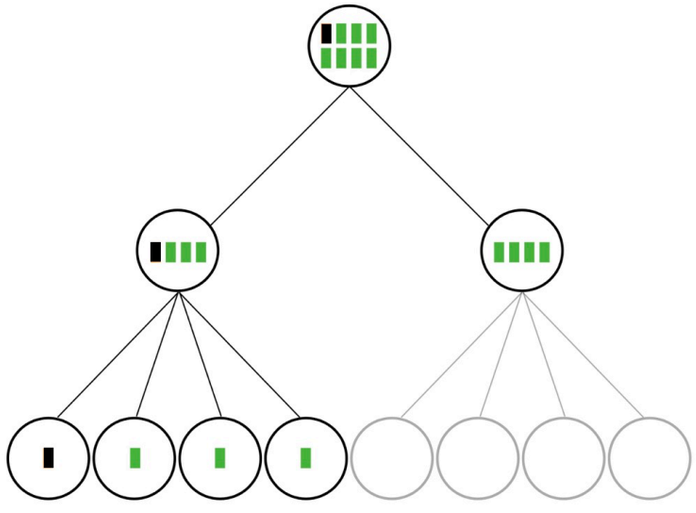
Program your simulation to test groups of 8 for a disease with an infection rate of 2% . For 1000 people (where 20 of them (2%) are infected and 980 are infection-free), you would make 125 groups of 8 samples each and simulate the tests. Based on the expected values (see the article) we expect the results to be as follows:

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

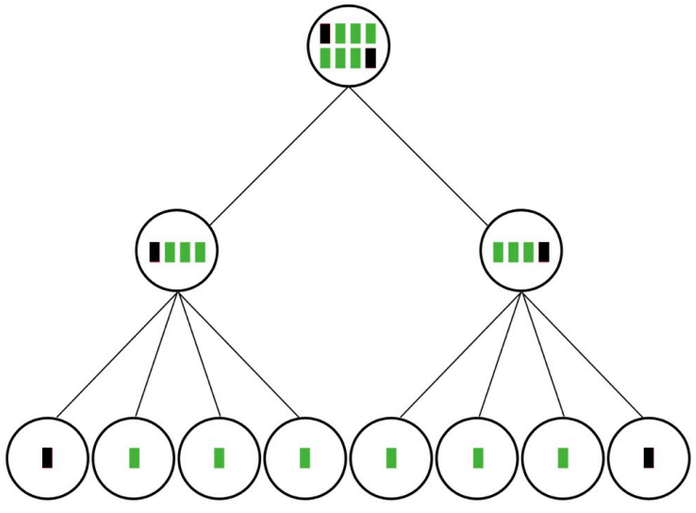
The output of your simulation should match or be very close to those values. Once you have that working, try it with 10,000 and 100,000 and 1M people, still in groups of 8, still with a 2% infection rate, still with 100% accurate tests. The larger the population you test, the closer you should get to the statistically expected values.



Case (1) : no infections, 1 test



Case (2) : 1 infection, 7 tests



Case (3) : 2+ infections, 11 tests (worst case)

Simulating the Testing

A Semester-long Project in Java or C++

Program your simulation to test groups of 8 for a disease with an infection rate of 2% . For 1000 people (where 20 of them (2%) are infected and 980 are infection-free), you would make 125 groups of 8 samples each and simulate the tests. Based on the expected values (see the article) we expect the results to be as follows:

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

The output of your simulation should match or be very close to those values. Once you have that working, try it with 10,000 and 100,000 and 1M people, still in groups of 8, still with a 2% infection rate, still with 100% accurate tests. The larger the population you test, the closer you should get to the statistically expected values.

Semester Project Results:

- 17 out of 20 students finished the project.
- Among those who finished…
  - mean grade was 98.5%.
  - All students were very good. Some students were exceptional.

Simulating the Testing

A Semester-long Project in Java

100% testing accuracy
8 people per pool
2% infection rate

- Pool support class combines Nodes and Binary Trees with a form of identification, *x* participants, and either positive or negative status as detected through testing.
- Left and Right pointer pools are only non-null if the original pool is positive.
- Population support class contains an ArrayList of 0's (healthy) and 1's (infected) individuals based on the infection rate

```
// Holds the original index of the Pool
// in the Level 1 ArrayList
int id;

// Holds "L" or "R" for Left or Right
// if this pool is a sub-Pool,
// otherwise is empty
String subID;

// Pointers to the left and right side of the pools
// May not need to be created if the pool is not
// positive or this pool is already a sub-pool.
// Use setSubPools to create both left and right
// versions of this pool.
Pool left;
Pool right;

// It is dangerous to make an assumption about the
// infection of a pool even with a low infection rate.
// We assume neither positive nor negative to be safe.
boolean positive;
ArrayList<Integer> group = new ArrayList<>();
  .
  .
  .
```

Simulating the Testing

A Semester-long Project in Java

The testing sequence tracks a lot of data.

For Statistics and Analytics

```
/* ALL POOL STATISTICS */
int originalInfected = 0;
int originalClear = 0;

int subPoolInfected = 0;
int subPoolClear = 0;

int infectedPeople = 0;
int clearPeople = 0;

/* TEST STATISTICS */
int level1Tests = 0;
int level2Tests = 0;
int individualTests = 0;

int case1 = 0; // No infections
int case2 = 0; // Single infection (1 sub-pool)
int case3 = 0; // 2+ infections (both sub-pools)
```

For Testing

```
// Testing accuracy in terms of decimal
// (for mathematical purposes)
double testAccuracy = 1;

// Temporary Pool that we use to construct
// the ArrayList of Pools
Pool tempPool = new Pool();

// All original pools of 8
ArrayList<Pool> populationPools = new ArrayList<>();

// Holds the id (that matches the index to
// populationPools)
// It is less computationally expensive to index into
// the original list than iterate over the whole array
// for low infection rates.
ArrayList<Integer> infectedPoolIDs = new ArrayList<>();
```

Simulating the Testing

A Semester-long Project in Java

100% testing accuracy
8 people per pool
2% infection rate

- Optimizes the pool size for uniform groups
- Testing sequence has three stages corresponding to tree diagrams
  - Level 1 runs the `for` loops testing the original pools, stores those found positive, and builds sub-pools
  - Level 2 runs a similar sequence, testing both left and right Pools, pushing all positives to individual.
  - Level 3 increments infection counter when located and stores ID of the sub-pool and attaches the index within the list.

Level 1 Testing

```
for(int i = 0; i < populationPools.size(); i++) {
  Pool currentPool = populationPools.get(i);

  if(currentPool.getGroup().contains(1)) {
    . . .
    currentPool.setPositive();
    currentPool.setSubPools();
    infectedPoolIDs.add(currentPool.getID());
  } else {
     currentPool.setNegative();
  }
  level1Tests++;
}
.
.
.
```

Simulating the Testing

A Semester-long Project in Java

This was run for varying populations, with 100% testing accuracy, 8 people per pool, and a 2% infection rate.

(1) Case 1: No infections

(2) Case 2: One infection

(3) Case 3: 2+ infections

Population	Case 1	Case 2	Case 3	Lvl 1 Tests	Lvl 2 Tests	Lvl 3 Test	**Total Tests**
1,000	104	29	2	125	42	92	**259**
10,000	1,067	172	11	1,250	366	776	**2,392**
100,000	10,612	1,818	70	12,500	3,76	7,832	**24,108**
1,000,000	106,951	17,915	690	125,000	37,210	77,180	**239,390**

Four Simulations Using Indicated Population Sizes

Simulating the Testing

A Semester-long Project in Java

Running the basic simulation multiple times, we can average the case occurrence rates and compare to the expected rates.

Population	Number of Runs	Case 1 Averages	Case 2 Averages	Case 3 Averages
1,000	29	106.2414 ≈ 0.849931	18.24138 ≈ 0.145931	0.517241 ≈ 0.004138
10,000	24	1063.333 ≈ 0.850667	178.75 ≈ 0.143	7.916667 ≈ 0.006333
100,000	24	10,641.54 ≈ 0.851323	1,781.667 ≈ 0.142533	76.79167 ≈ 0.006143
1,000,000	24	106345.6 ≈ 0.850765	17901.38 ≈ 0.143211	753.0416667 ≈ 0.006024333
Expected		**0.85**	**0.14321**	**0.0060267**

Multi-Run Averages & Occurrence Rates

Simulating the Testing

Statistics vs. Simulation

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

Population	Number of Runs	Case 1 Averages	Case 2 Averages	Case 3 Averages
1,000	29	106.2414 ≈ 0.849931	18.24138 ≈ 0.145931	0.517241 ≈ 0.004138
10,000	24	1063.333 ≈ 0.850667	178.75 ≈ 0.143	7.916667 ≈ 0.006333
100,000	24	10,641.54 ≈ 0.851323	1,781.667 ≈ 0.142533	76.79167 ≈ 0.006143
1,000,000	24	106345.6 ≈ 0.850765	17901.38 ≈ 0.143211	753.0416667 ≈ 0.006024333
Expected		**0.85**	**0.14321**	**0.0060267**

Multi-Run Averages & Occurrence Rates

Simulating the Testing

Statistics vs. Simulation

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

This is spot-on. Yay math!

Population	Number of Runs	Case 1 Averages	Case 2 Averages	Case 3 Averages
1,000	29	106.2414 ≈ 0.849931	18.24138 ≈ 0.145931	0.517241 ≈ 0.004138
10,000	24	1063.333 ≈ 0.850667	178.75 ≈ 0.143	7.916667 ≈ 0.006333
100,000	24	10,641.54 ≈ 0.851323	1,781.667 ≈ 0.142533	76.79167 ≈ 0.006143
1,000,000	24	106345.6 ≈ 0.850765	17901.38 ≈ 0.143211	753.0416667 ≈ 0.006024333
Expected		**0.85**	**0.14321**	**0.0060267**

Multi-Run Averages & Occurrence Rates

Simulating the Testing

Statistics vs. Simulation

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

This is pretty good, but the stats are a little high.

Population	Number of Runs	Case 1 Averages	Case 2 Averages	Case 3 Averages
1,000	29	106.2414 ≈ 0.849931	18.24138 ≈ 0.145931	0.517241 ≈ 0.004138
10,000	24	1063.333 ≈ 0.850667	178.75 ≈ 0.143	7.916667 ≈ 0.006333
100,000	24	10,641.54 ≈ 0.851323	1,781.667 ≈ 0.142533	76.79167 ≈ 0.006143
1,000,000	24	106345.6 ≈ 0.850765	17901.38 ≈ 0.143211	753.0416667 ≈ 0.006024333
Expected		**0.85**	**0.14321**	**0.0060267**

Multi-Run Averages & Occurrence Rates

Simulating the Testing

Statistics vs. Simulation

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

This is wrong by 0.0056. Not bad, but the stats are too low. Why?

Population	Number of Runs	Case 1 Averages	Case 2 Averages	Case 3 Averages
1,000	29	106.2414 ≈ 0.849931	18.24138 ≈ 0.145931	0.517241 ≈ 0.004138
10,000	24	1063.333 ≈ 0.850667	178.75 ≈ 0.143	7.916667 ≈ 0.006333
100,000	24	10,641.54 ≈ 0.851323	1,781.667 ≈ 0.142533	76.79167 ≈ 0.006143
1,000,000	24	106345.6 ≈ 0.850765	17901.38 ≈ 0.143211	753.0416667 ≈ 0.006024333
Expected		**0.85**	**0.14321**	**0.0060267**

Multi-Run Averages & Occurrence Rates

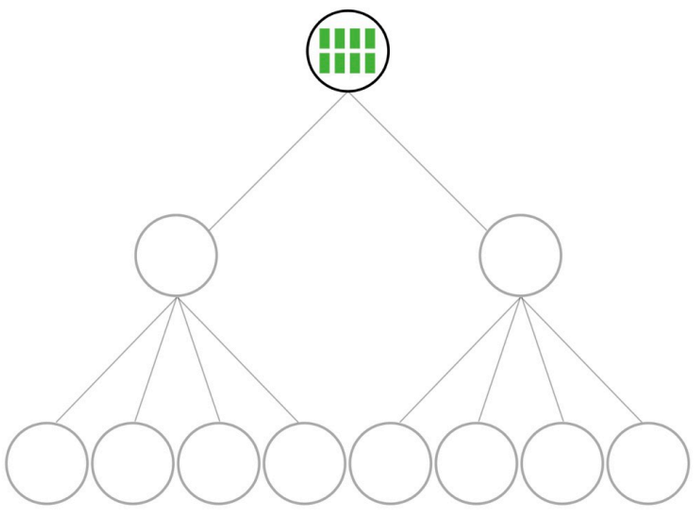
Simulating the Testing

Actually, it's not that simple.

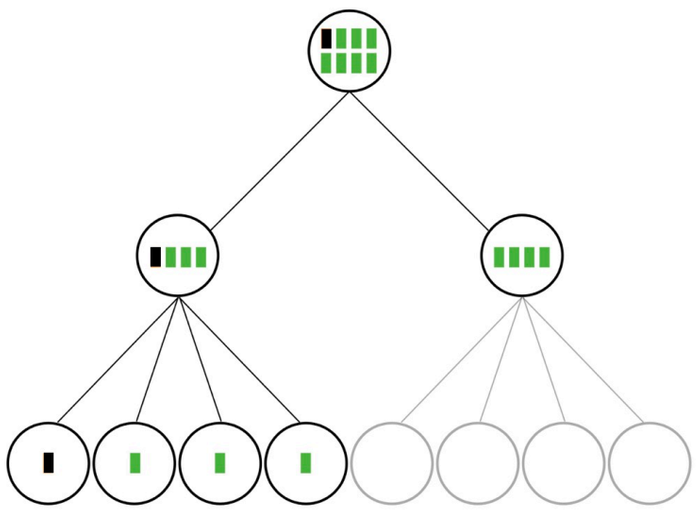
There are issues:

- The simplified example assumed selection **with** replacement (binomial distribution). But selection **without** replacement (hyper-geometric distribution) is what we're doing.
- There is overlap in Case 2 and Case 3. There can be multiple infections in the same sub-group. We assumed the worst-case in terms of test usage, which is why there are some differences.

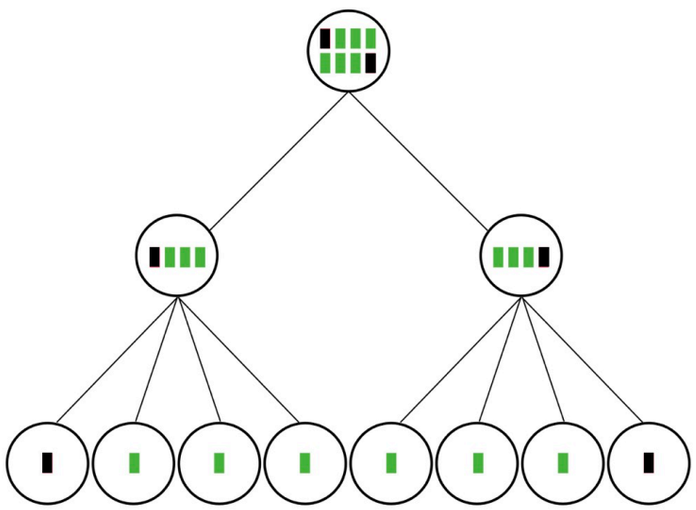
But it's worse than that. Because, even taking these issues into account . . .



Case (1) : no infections, 1 test



Case (2) : 1 infection, 7 tests



Case (3) : 2+ infections, 11 tests (worst case)

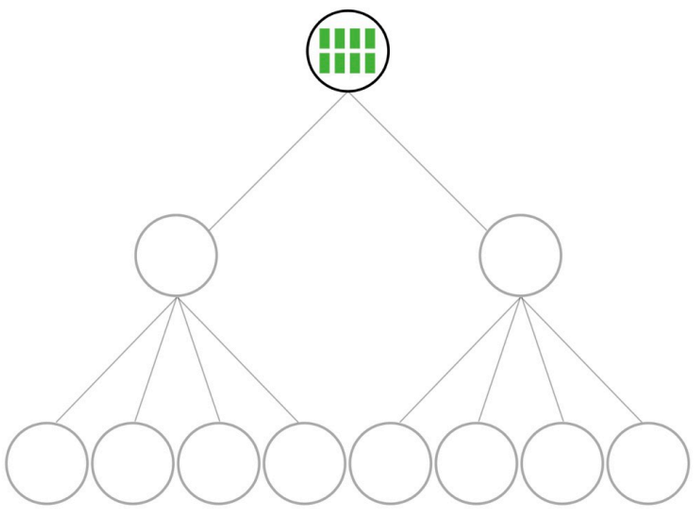
Simulating the Testing

Actually, it's not that simple.

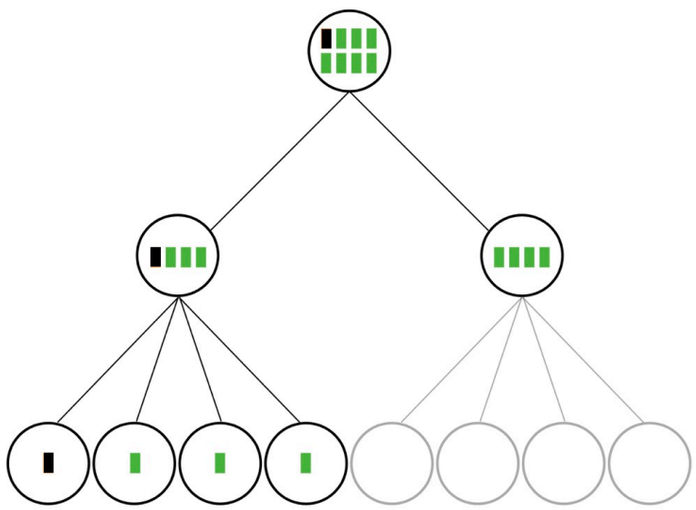
Test Accuracy is never realistically 100%

- Tests have a chance of reporting **False Positives** and **False Negatives**.
- The latter is most dangerous, since the pool is only tested once and the infected individual has potential to unknowingly spread infection.
- The more people contained in a pool, the higher the risk of producing false results.

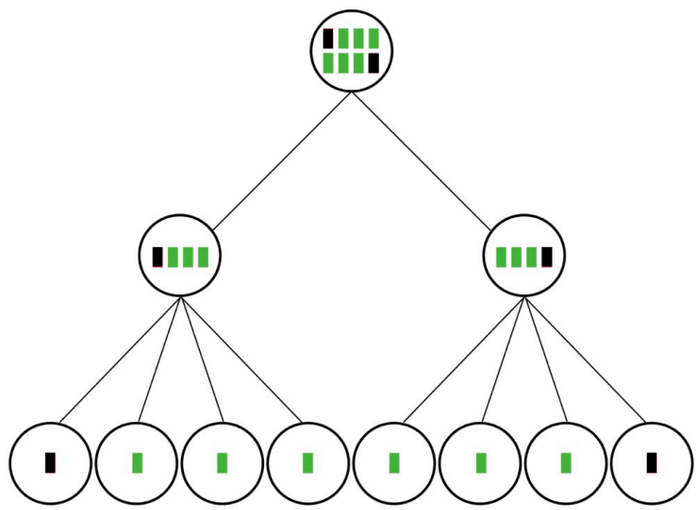
We needed to make it better.



Case (1) : no infections, 1 test



Case (2) : 1 infection, 7 tests



Case (3) : 2+ infections, 11 tests (worst case)

Background

Simulating the Testing

Testing the Simulation

Testing the Simulation

A Better Analysis

There are four conditional probability cases to track:

(1) *True Positive* $P(T^+ \mid D^+)$ is known as **Testing Sensitivity** and provides the possibility of testing positive given an individual is actually infected.

(2) *True Negative* $P(T^- \mid D^-)$ is known as **Testing Specificity** and provides the possibility of testing negative given an individual is actually not infected.

(3) *False Positive* $P(T^+ \mid D^-)$ provides the possibility of testing positive given an individual is actually not infected.

(4) *False Negative* $P(T^- \mid D^+)$ provides the possibility of testing negative given an individual is actually infected.

Testing the Simulation

A Better Analysis

This can be summarized as a *confusion matrix*, showing the occurrence rates we expect for each case.

Suppose, for example, an infection rate of 5%, with Sensitivity and Specificity = 98%

	INFECTED D^+	NOT INFECTED D^-	
T^+	$= (TA_{Sen})(IR)$ $= (0.98)(0.05)$ $=$ **0.049** TP	$= (1 - TA_{Spec})(1 - IR)$ $= (1 - 0.98)(1 - 0.05)$ $=$ **0.019** FP	$P(T^+) =$**0.068**
T^-	$= (1 - TA_{Sen})(IR)$ $= (1 - 0.98)(0.05)$ $=$ **0.001** FN	$= (TA_{Spec})(1 - IR)$ $= (0.98)(1 - 0.05)$ $=$ **0.931** TN	$P(T^-) =$ **0.932**
	$P(D^+) =$ **0.05**	$P(D^-) =$ **0.95**	**1**

Trial Example Expected Population Occurrence Rates

Testing the Simulation

A Better Simulation

- Completely redesigned the structure and added additional formatting and statistical classes that contained hand-made functions.
- Restructured to allow for pools of size 1 (no pooled testing), which required some index manipulation for a `for` loop to run correctly no matter the architecture.
- Customizable testing structure can allow for different stages such as 8-4-2-1, 8-4-1, or 12-1, which required additional case handling to check if the number of declared stages was the same as the maximum number of stages possible and other restrictive qualities.
- Handles a lot of new cases, but computationally expensive testing algorithm
  - between $O(n^2)$ and $O(n^3)$

Testing the Simulation

A Better Simulation

- Pooled Testing Accuracy and Pool Dilution
  - Originally computed fully healthy pool occurrence rate as Testing Specificity to the power of the pool size.
  - Ignored real-world sample dilution, where infected samples are diluted by healthy samples and become undetectable in large batches.
    - Incredibly complex!
    - Even though there are formulas for calculating the viral load of a mixed fluid sample, they do not show how to calculate the probability of a pool testing positive or negative.

Testing the Simulation

A Better Simulation

- Pooled Testing Accuracy and Pool Dilution
  - Originally computed fully healthy pool occurrence rate as Testing Specificity to the power of the pool size.
  - Ignored real-world sample dilution, where infected samples are diluted by healthy samples and become undetectable in large batches.
    - Incredibly complex!
    - Even though there are formulas for calculating the viral load of a mixed fluid sample, they do not show how to calculate the probability of a pool testing positive or negative.

(This will be important later.)

Testing the Simulation

New and Improved Adaptive Infection Simulator

- All probabilities between 0 and 1.
- Sensitivity and Specificity, Infection Rate, Population Size, and number of Levels (Stages) taken from the user.
- Infection Rate must be < 50%
- Testing measures must be > 75%
- Customizable pool size, optimized by mathematical constraints otherwise.
- Testing architecture can be customized for desired pool-splitting.

```
numInfected =
(int)Math.round(populationSize*infectionRate);

for (int i = 0; i < populationPools; i++) {
   if (i < numberInfected) {
      simPopulation.add(1);
   } else {
      simPopulaiton.add(0);
   }
}
Collections.shuffle(simPopulation);
```

- ArrayList stores population using 0's (healthy) and 1's (infected)
  - To ensure we have exactly the number of infections expected in the population, we multiply the population size by the infection rate to get the number of 1's added to the ArrayList, then fill the remainder with 0's, and (Knuth) shuffle.

Testing the Simulation

Challenges in comparing it to real data

Marist used a protocol clearly not designed by computer scientists.

- Pools (usually) of size 12
  - Fine, we can adjust our simulation.
- If an infection is found then go directly to individual tests.
  - Fine, we can adjust our simulation again.
- Protocol imposed by testing lab, not us.
  - (They didn't even ask.)

Infection rate varied over time.

- Need to focus the timeframe to make a meaningful comparison.
- Used two-week window: March 15-28, 2021
- Later in the semester, vaccinated people could abstain. How to model that?

Data includes asymptomatic pooled testing, **not** symptomatic testing.

Testing the Simulation

Real Data

(*Shared with permission.*)

We compared actual data from the second half of March 2021…

```
-- Distinct tests this period
SELECT count(distinct studentID)
FROM processedLabResults
WHERE collectionTime between '2021-03-15' and '2021-03-28';
-- 3707 Distinct tests this period

-- Individuals in a positive pool this period
SELECT count(studentID)
FROM processedLabResults
WHERE collectionTime between '2021-03-15' and '2021-03-28'
  and SARSCOV = 'POOL POSITIVE';
-- 748 Individuals in a positive pool this period
--     (Each pool is supposed to be 12 people, so that's about 62 positive pools)

-- Positive (distinct) individual tests this period
SELECT count(distinct studentID)
FROM processedReflexResults
WHERE collectionTime between '2021-03-15' and '2021-03-28'
  and SARSCOV = 'REFLEX POSITIVE';
-- 35 Positive (distinct) individual tests this period

-- More Positive individual tests this period, NOT in the processedReflexResults table (blame the lab)
SELECT count(distinct studentID)
FROM processedLabResults p
WHERE upper(p.reflexTestResult) like '%POSITIVE%'
  and p.collectionTime between '2021-03-15' and '2021-03-28'
-- 6 More Positive individual tests this period, NOT in the processedReflexResults table

-- 35+6=41 individual positives in 3707 distinct individuals = infection rate of 1.1% this period.
```

Testing the Simulation

A Real Comparison

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
INITIALIZING.....

PLEASE ENTER PERCENTAGES AS DECIMALS LIKE 0.05 = 5%
ALL INPUT VALUES ARE VALIDATED BEFORE PROCEEDING

Enter the infection rate: 0.011
Infection Rate Accepted

Enter the testing sensitivity (accuracy, TPR): 1.0
Testing Accuracy Accepted

Enter the testing specificity (TNR): 1.0
Testing Specificity Accepted

Enter the population size as an integer: 3707
Population size Accepted

-------------------------------------------------------
Individual True Positive:  0.011
Individual False Positive: 0.000
Individual True Negative:  0.989
Individual False Negative: 0.000
-------------------------------------------------------
Individual Testing Negative Probability: 0.989
Individual Testing Positive Probability: 0.011
-------------------------------------------------------
```

100% sensitivity and specificity

Testing the Simulation

A Real Comparison

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
INFECTING THE POPULATION.........
Based on the Infection Rate and Testing Accuracy,
the largest Pool size for efficient Pooled Testing is 11

Would you like to override the Pool size? Y/N: y
Enter new Pool Size: 12

The Pool can be split a maximum of 4 times,
so the maximum number of testing levels is 5
-----------------------------------------------------
1 TESTING LEVELS = INDIVIDUAL TESTING
2 TESTING LEVEL = ORIGINAL POOLS > INDIVIDUAL TESTING
3 TESTING LEVELS = ORIGINAL POOLS > SUB-POOLS > INDIVIDUAL TESTING
4 TESTING LEVELS = ORIGINAL POOLS > SUB-POOLS > SUB-SUB-POOLS > INDIVIDUAL TESTING

-----------------------------------------------------
Please enter the number of desired testing levels: 2
Number of Testing Levels Accepted
```

Lab protocol

Testing the Simulation

A Real Comparison

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
---------------------------------------
Real Data
---------------------------------------
Infection rate  = 1.1% this period
Population size = 3707 this period

748 Individuals in a positive pool
 62 positive pools

35+6=41 Positive individual tests
```

```
-----------------------------------------------------------
LEVEL 1 METRICS - 309 Pools
-----------------------------------------------------------
| True Positive:  40 | False Positive:   0 | Total =   40 > 0.1294
| False Negative:  0 | True Negative: 269 | Total = 269 > 0.8706

| P(D+|T+): 1.0 | P(D-|T+): 0.0
| P(D+|T-): 0.0 | P(D-|T-): 1.0
```

```
LEVEL 2 METRICS (INDIVIDUAL TESTING) - 480 Individual Tests
-----------------------------------------------------------
| True Positive:  41 | False Positive:   0 | Total =   41 > 0.0854
| False Negative:  0 | True Negative: 439 | Total = 439 > 0.9146

| P(D+|T+): 1.0 | P(D-|T+): 0.0
| P(D+|T-): 0.0 | P(D-|T-): 1.0

-----------------------------------------------------------
Number of tests used: 789
-----------------------------------------------------------
```

Very accurate.
This is a single simulation run.
We did more and computed the average.

Testing the Simulation

A Real Comparison - average over multiple runs

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
---------------------------------------
Real Data
---------------------------------------
Infection rate  = 1.1% this period
Population size = 3707 this period

748 Individuals in a positive pool
 62 positive pools

35+6=41 Positive individual tests
```

```
-----------------------------------------------------------
LEVEL 1 METRICS - 309 Pools
-----------------------------------------------------------
| True Positive: 38 | False Positive:  0 | Total =  38
| False Negative: 0 | True Negative: 271 | Total = 271

| P(D+|T+): 1.0 | P(D-|T+): 0.0
| P(D+|T-): 0.0 | P(D-|T-): 1.0
```

```
LEVEL 2 METRICS (INDIVIDUAL TESTING) - 480 Individual Tests
-----------------------------------------------------------
| True Positive: 41 | False Positive:  0 | Total =  41
| False Negative: 0 | True Negative: 415 | Total = 415

| P(D+|T+): 1.0 | P(D-|T+): 0.0
| P(D+|T-): 0.0 | P(D-|T-): 1.0

-----------------------------------------------------------
Number of tests used: 789
-----------------------------------------------------------
```

Still accurate.
No significant changes.

The real data has more positive pools but the same number of individual results. Why?
Grouping? Data quality?
Test accuracy issues (e.g., pool dilution)?

Let's run another simulation, this time with 95% test accuracy.

Testing the Simulation

A Real Comparison

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
INITIALIZING.....

PLEASE ENTER PERCENTAGES AS DECIMALS LIKE 0.05 = 5%
ALL INPUT VALUES ARE VALIDATED BEFORE PROCEEDING

Enter the infection rate: 0.011
Infection Rate Accepted

Enter the testing sensitivity (accuracy, TPR): 0.95
Testing Accuracy Accepted

Enter the testing specificity (TNR): 0.95
Testing Specificity Accepted

Enter the population size as an integer: 3707
Population size Accepted

-----------------------------------------------------
Individual True Positive:  0.0104
Individual False Positive: 0.0495
Individual True Negative:  0.9396
Individual False Negative: 6.0E-4
-----------------------------------------------------
Individual Testing Negative Probability: 0.9402
Individual Testing Positive Probability: 0.0599
-----------------------------------------------------
```

95% sensitivity and specificity

Testing the Simulation

A Real Comparison

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
INFECTING THE POPULATION.........
Based on the Infection Rate and Testing Accuracy,
the largest Pool size for efficient Pooled Testing is 4

Would you like to override the Pool size? Y/N: y
Enter new Pool Size: 12

The Pool can be split a maximum of 4 times,
so the maximum number of testing levels is 5
-----------------------------------------------------
1 TESTING LEVELS = INDIVIDUAL TESTING
2 TESTING LEVEL = ORIGINAL POOLS > INDIVIDUAL TESTING
3 TESTING LEVELS = ORIGINAL POOLS > SUB-POOLS > INDIVIDUAL TESTING
4 TESTING LEVELS = ORIGINAL POOLS > SUB-POOLS > SUB-SUB-POOLS > INDIVIDUAL TESTING

-----------------------------------------------------
Please enter the number of desired testing levels: 2
Number of Testing Levels Accepted
```

Same lab protocol

Testing the Simulation

A Real Comparison

We compared actual data from the second half of March 2021 with predictions made by our simulation.

```
---------------------------------------
Real Data
---------------------------------------
Infection rate  = 1.1% this period
Population size = 3707 this period

748 Individuals in a positive pool
 62 positive pools

35+6=41 Positive individual tests
```

```
-----------------------------------------------------------
LEVEL 1 METRICS - 309 Pools
-----------------------------------------------------------
| True  Positive: 20 | False Positive: 115 | Total = 135 > 0.4369
| False Negative: 21 | True  Negative: 153 | Total = 174 > 0.5631

| P(D+|T+): 0.1481 | P(D-|T+): 0.8519
| P(D+|T-): 0.1207 | P(D-|T-): 0.8793
```

```
LEVEL 2 METRICS (INDIVIDUAL TESTING) - 1619 Individual Pools
-----------------------------------------------------------
| True  Positive: 20 | False Positive:   78 | Total =   98 > 0.0605
| False Negative:  0 | True Negative: 1521 | Total = 1521 > 0.9395


| P(D+|T+): 0.2041 | P(D-|T+): 0.7959
| P(D+|T-): 0.0    | P(D-|T-): 1.0
```

Only 20 true positives is worrying.

21 false negatives is terrifying.

```
-----------------------------------------------------------
Number of tests used: 1928
-----------------------------------------------------------
```

Testing the Simulation

Future Work

For us

- Refine these models.
- Compare to more real data.
- Write more papers.

Testing the Simulation

Future Work

For us

- Refine these models.
- Compare to more real data.
- Write more papers.

For **you**

- Build on this work.
  Code available on GitHub: https://github.com/labouseur/CovidInTheClassroom
- Write papers.

Testing the Simulation

Future Work

For us

- Refine these models.
- Compare to more real data.
- Write more papers.

For **you**

- Build on this work.
  Code available on GitHub: https://github.com/labouseur/CovidInTheClassroom
- Write papers.

Thank you!

- Hope.Neveux1@Marist.edu
- Alan.Labouseur@Marist.edu

Questions? Suggestions?

Oh, so you want to

see some code? Sure…

We can do that!

Testing the Simulation

New and Improved Adaptive Infection Simulator

- Store pools in an *ArrayList*
- Use a `for` loop running for the number of levels (stages), where each has their own case counters
- Inner loop for the number of pools we want to test in the subsequent stage
- Randomly generate a number between 0 and 1 for each pool
- Check if the pool contains an infection, which reroutes to the Positive or Negative sections of the sequence
- If the randomly generated number falls below the Sensitivity (if infection) or Specificity (if clear) raised to the power of the pool size, we have the **True** Positive or Negative, otherwise it's **False**.

```
testPools = originalPools
for # of testing stages {
|  infected = temp for pools to test in next stage
|  initialize counters for the 4 cases
|  for # testPools {
|  |  infectedVal = randomly generated value
|  |  if pool doesn't actually have a 1 (no infection)
|  |  |  if infectedVal <= Specificity^k
|  |  |  |  Set Negative, increment TN
|  |  |  else
|  |  |  |  Set Positive, increment FP, add to pos. array
|  |  |  |  if pools are individual stage
|  |  |  |  |  Build individual Pools, add all to infected
|  |  |  |  else if Pool isn't already individual
|  |  |  |  |  Build Left and Right Pools, add to infected
|  |  |  |  end if
|  |  |  endif
|  |  else
|  |  |  if infectedVal <= Sensitivity^k
|  |  |  |  Set Positive, increment TP, add to pos. array
|  |  |  |  if pools are individual stage
|  |  |  |  |  Build individual pools, add all to infected
|  |  |  |  else if the pools isn't already individual
|  |  |  |  |  Build Left and Right Pools, add to infected
|  |  |  |  end if
|  |  |  else
|  |  |  |  Set Negative, increment FN, add to FN array
|  |  |  end if
|  |  end if
|  |  increment testsUsed
|  end for
|  overwrite testPools, clear infected, store metrics
end for
```

Testing the Simulation

New and Improved Adaptive Infection Simulator

- Pools registering positive are split into halves or individuals and added to a temporary container
- If we aren't dealing with individual testing in the next stage, the Left Pool gets the 1st half with the same ID-*L* and the Right Pool gets the 2nd half with the same ID-*R*
- If we are dealing with individual testing in the next stage, we assign each person to a pool of their own, holding the same ID-*original index number*
- Store false negatives, since those tell us which pools were missed by the simulator and at what stage
- Every time a pool is checked we increment a test counter.

```
testPools = originalPools
for # of testing stages {
|  infected = temp for pools to test in next stage
|  initialize counters for the 4 cases
|  for # testPools {
|  |  infectedVal = randomly generated value
|  |  if pool doesn't actually have a 1 (no infection)
|  |  |  if infectedVal <= Specificity^k
|  |  |  |  Set Negative, increment TN
|  |  |  else
|  |  |  |  Set Positive, increment FP, add to pos. array
|  |  |  |  if pools are individual stage
|  |  |  |  |  Build individual Pools, add all to infected
|  |  |  |  else if Pool isn't already individual
|  |  |  |  |  Build Left and Right Pools, add to infected
|  |  |  |  end if
|  |  |  endif
|  |  else
|  |  |  if infectedVal <= Sensitivity^k
|  |  |  |  Set Positive, increment TP, add to pos. array
|  |  |  |  if pools are individual stage
|  |  |  |  |  Build individual pools, add all to infected
|  |  |  |  else if the pools isn't already individual
|  |  |  |  |  Build Left and Right Pools, add to infected
|  |  |  |  end if
|  |  |  else
|  |  |  |  Set Negative, increment FN, add to FN array
|  |  |  end if
|  |  end if
|  |  increment testsUsed
|  end for
|  overwrite testPools, clear infected, store metrics
end for
```